We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

13

Evolutionary Based Controller Design

Ivan Sekaj
Slovak University of Technology,
Faculty of Electrical Engineering and Information Technology,
Bratislava,
Slovak Republic

1. Introduction

Control design methods have to respect various requirements imposed on the performance of controlled systems. Controllers often include many design parameters and their various constraints. The search/optimisation process may be complicated, discontinuous or non-convex, and often some analytical methods are not able to yield satisfactory results. Opposite to this, evolution-based search techniques are able to generate new control laws and non-intuitive solutions as well. Without loss of generality, let us consider the use of genetic algorithm (GA), which is one of the most frequently used evolutionary techniques. Recently, genetic algorithms have been applied in process control to solve a wide range of optimisation and design problems. This chapter is focused on the design of controllers for continuous-time systems.

Let us classify the known approaches, which are using evolutionary algorithms (EA) in two groups. In the first group the EA's are used as a powerful optimisation/search tool in analytically formulated control design methods. The parameters of controllers (or any dynamic system) are designed analytically, based on the stability or required system performance (Kawabe et al., 1996; Krohling & Rey, 2001; Man, 2001; Sekaj & Veselý, 2005). The second group of methods applies simulation-based closed-loop time response evaluation of the model (Herrero et al., 2002; Khatib, 1999; Lewin, 2005; Mitsukura et all, 1999; Sweriduk et al., 1999; Yang, 2005). A multi-objective approach that comprises 7 different objectives including both analytically formulated and time-response performance based ones is presented in (Molina-Cristóbal, 2005).

If the design task includes not just searching for parameters of a prespecified control structure but also searches for the internal structure itself, an extension to this approach is possible using the Genetic programming (Koza, 1992; Banzhaf et al., 1999; Koza et al., 2000; Sekaj et al., 2005; Sekaj & Perkacz, 2007). To optimise controller structure also a hierarchical GA can be used (Man at al., 2001). In (Grosman & Lewin, 2005) Genetic programming has been used to generate the Lyapunov functions. Survey of evolutionary-based control system designs is e.g. in (Fleming & Purshouse; Lewin, 2005).

In this chapter a straightforward incorporating of the simulation-based closed-loop time response evaluation in the evolutionary algorithm is presented. The proposed approach deals with a direct Evolutionary-based search/optimisation in the controller parameter space combined with extensive computer simulations of the designed closed-loop system

Source: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos,
ISBN 978-953-307-008-7, pp. 572, October 2009, I-Tech, Vienna, Austria

(Sekaj, 1999; Sekaj et al., 2002; Sekaj, 2003; Sekaj et al., 2005; Sekaj & Perkacz, 2007; Sekaj 2005). Thus, the simulation is an essential part of the minimised objective function. It will be shown that according to this approach, the design of optimal parameters for a dynamic system is transformed into a "conventional" n-dimensional optimisation problem. In section 2, the design principle is described and the use of various performance indices is discussed. In the section 3 robust controller design methods are proposed. A multi-objective design approach is discussed in section 4, where multiple objectives are considered in the controller design procedure. Finally, controller internal structure design and its parameters using genetic programming is proposed in section 5.

2. Controller design

2.1 The design principle

As already mentioned, the control design objective is to provide required static and dynamic behaviour of the controlled process, usually represented in terms of the well-known performance measures: maximum overshoot, settling time, decay rate, steady state error or various integral performance indices (Dorf, 1990; Kuo, 1991; and others).

Without loss of generality let us consider a simple feedback loop (Fig. 1) where y is the controlled variable, u is the control, r is the reference and e is the control error ($e=r-y$). Let an appropriate simulation model of the controlled object be available. The closed-loop performance will be assessed using the simple integral performance index "integral of absolute control error" defined as

$$I_{AE} = \int_0^T |e(t)| dt \tag{1}$$

where T is the simulation time. The discrete form of this performance index is

$$I_{AE} = \sum_{k=1}^{N} T_{s,k} |e_k| \tag{2}$$

where $T_{s,k}$ is the simulation step size and N is the number of simulation steps.

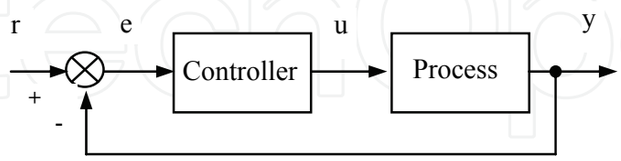

Fig. 1. Simple feedback control loop

The controller design is actually an optimisation task - search for such controller parameters from the defined parameter space that minimise the performance index (1) or (2). The cost function (fitness) is a mapping $R^n \rightarrow R$, where n is the number of designed controller parameters. The cost function evaluation consists of two steps. The first step is the closed-loop time response simulation, the second one is the performance index evaluation. Note, that when designing complex, multi-input multi-output (MIMO) control structures or the

other controller types (fuzzy controllers, neuro-controllers, etc.) the dimension n of the search space may be quite large (more than tens or even hundreds).
Graphical representation of a simple cost function corresponding to a simple PI controller design is in Fig. 2. Each point of the surface $G_{IAE}=f(P,I)$ is a result of a simulation and the performance index (1) evaluation. The optimal PI controller parameters are represented by the co-ordinates of the surface global minimum. This simple 2-D search problem can also be solved using conventional (deterministic) enumerative search/optimisation techniques. However, in case of more complex control structures comprising many parameters, the dimension of the search space is growing and the use of such methods may be no more feasible due to high computational requirements. Here, the evolutionary-based techniques, in our case the genetic algorithms can successfully be used.

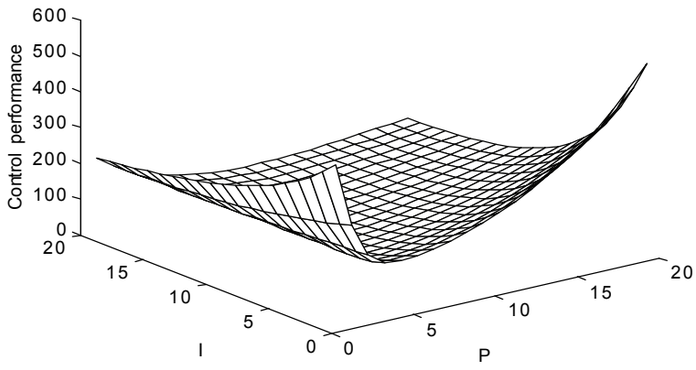

Fig. 2. Graphical representation of a cost function with the I_{AE} performance index (1) for a particular closed-loop under a PI controller

2.2 Genetic algorithm

Genetic algorithms (Goldberg, 1989; Michalewicz, 1996; Eiben & Smith, 2003; Man et al., 2001) deals with a population consisting of a set of chromosomes. Each chromosome represents a potential solution and has the form of a linear string of numbers; in our case, items of a chromosome (genes) correspond to the designed controller parameters. As the controller parameters are real-number variables and the number of the searched parameters can be large, real-coded chromosomes have been considered.
Without loss of generality consider a PID controller, which control law in time domain description is as follows

$$u(t) = Pe(t) + I\int e(t)dt + D\frac{de(t)}{dt} \tag{3}$$

where $P \in R$, $I \in R$, $D \in R$ are the proportional, integral and derivative gains, respectively. In this case a chromosome can be represented in the form $ch=\{P,I,D\}$. For any other controller type with parameters $c_1, c_2, \ldots, c_q$ the corresponding chromosome would be a string

$$ch=\{c_1, c_2, \ldots, c_q\}.$$

A general scheme of a GA can be described as follows:

1. Initialisation of the population of chromosomes (generating a set of random chromosomes).
2. Evaluation of the cost function (fitness) for all chromosomes.
3. Selection of parent chromosomes.
4. Crossover and mutation of parents → children.
5. Creation of a new population consisting of new children and selected members of the old population including the best individual from the current population.
6. Testing terminating conditions, jump to the Step 2 or End.

The block scheme describing the controller design principle is shown in Fig. 3.

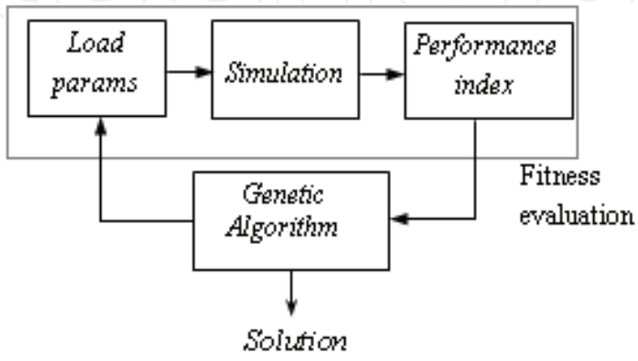

Fig. 3. Block scheme of the GA-based controller design

Example in Fig. 4 shows evolution of a PID via minimisation of the criterion (1). After a certain number of generations the best solution from the current population (its closed-loop step response) has been plotted. As after 100 generations the solution has not changed considerably, the GA run could have been terminated. The cost function convergence during three independent GA-runs (cost function of the currently best individual of the population vs. generation number) is depicted in Fig. 5.

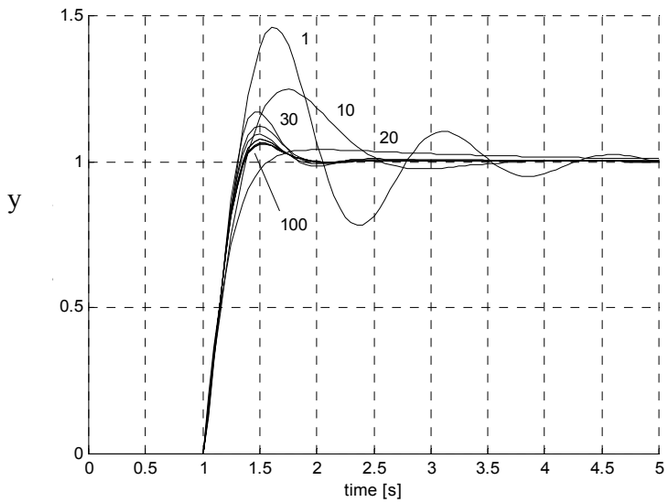

Fig. 4. Evolution of the PID controller parameters after 1, 10, 20, 30 and 100 generations

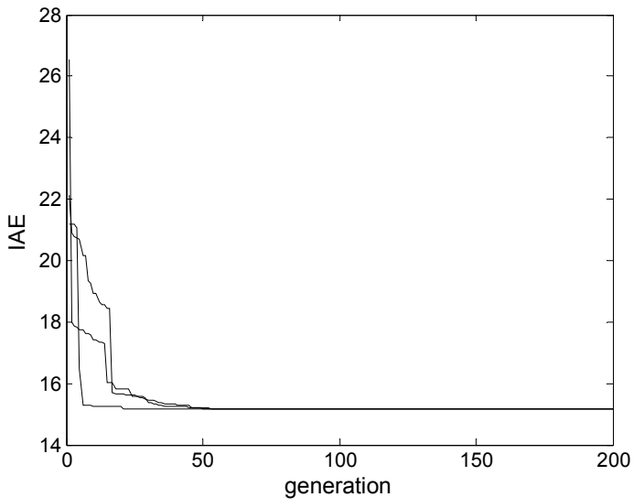

Fig. 5. Cost function convergence of the PID controller design procedure for three independent GA-runs for population size 30 individuals.

Note, that the controller design for a complex system can become a multi-modal and a time-consuming problem, where often a good sub-optimal solution is sufficient. The answer to the question about the convergence rate of the controller design procedure is similar as in case of other numerical GA-based search/optimisation problems. It depends on the search space size and dimension and on the GA performance.

2.3 Choice of the performance index

Consider that a GA has found the optimal (sub-optimal) solution in the user-defined search space of controller parameters. The choice of the performance index has a fundamental influence on the closed-loop dynamics. Normally, using (1) or (2) brings about fast control responses with small overshoots between 2-5% (Fig. 4). For various objectives different performance indices can be used (Sekaj, 1999; Sekaj, 2005).

If it is necessary to reduce overshoot or to damp oscillations, it is recommended to insert in the integral additional terms, which include absolute values of the first or also the second derivatives of the control error

$$J = \int_0^T \left(\alpha |e(t)| + \beta |e'(t)| + \gamma |e''(t)| \right) dt \tag{4}$$

and to increase β and γ with respect to α, where α, β, γ are weight coefficients. Note, that the control error derivatives can be replaced with the absolute values of output derivatives ($|y'(t)|$, $|y''(t)|$). In the discrete case the integral is replaced by the sum and the derivatives by the differences. Good results can be obtained also using the performance index in the form

$$J = \alpha \eta + (1 - \alpha) t_s \tag{5}$$

where η is the overshoot, t_s is the settling time and $0 < \alpha < 1$ is the weight coefficient. Tracking the reference step response $y_r(t)$ is achieved via minimising

$$J = \int (y_r(t) - y(t))^2 dt \tag{6}$$

Control energy minimisation can be achieved using performance indices of the type

$$J = \int (\alpha e^2(t) + (1-\alpha)u^2(t))dt \tag{7}$$

where u is the control variable. A universal performance index, which combines some above criterions is as follows

$$J = \int_0^T \left(|e(t)| + a|e'(t)| + b|u(t)| + c|u'(t)| \right) dt \tag{8}$$

where e' is the control error derivative, u is the control variable and u' is its derivative. This performance index includes oscillation damping (increasing a), minimisation of the absolute value of the control signal u (increasing b) and minimisation of control signal change u' (increasing c). In Fig. 6 to Fig. 13 various configurations of parameters *a, b*, and *c* in (8) and their influence of the closed-loop time response *y* and control value *u* are depicted.
In case of designing multi-input multi-output systems the design procedure is analogical, but for each output a particular (i-th) part of the cost function with some weight is considered (9).

$$J = \sum_{i=1}^{N} w_i J_i \tag{9}$$

Remark about the closed-loop stability: Due to the applied performance index minimisation, closed-loop stability is an implicit attribute of the controller design process. Unstable chromosomes are eliminated during the evolution because of high values of performance index and thus the solution is directed into the region of stable parameters. However if necessary, it is possible to include some stability test in each fitness evaluation. Cost function values of unstable individuals can additionally obtain high penalty values.

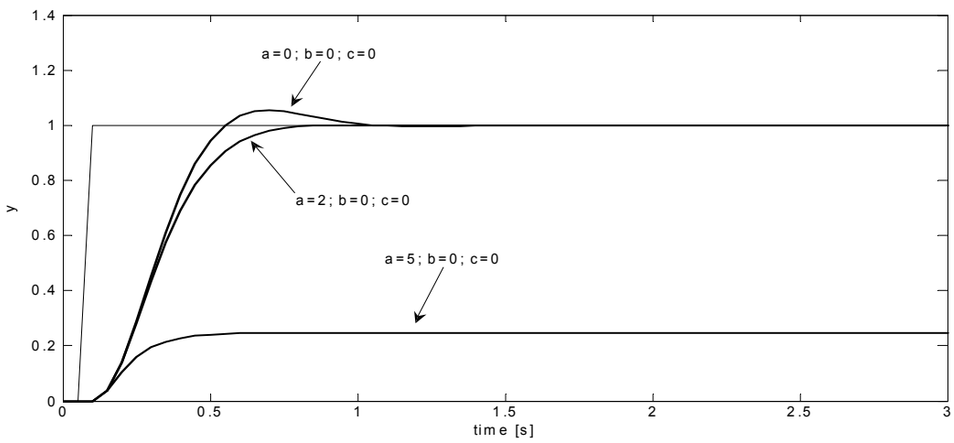

Fig. 6. Closed-loop time responses for various values of the parameter *a* in (8)

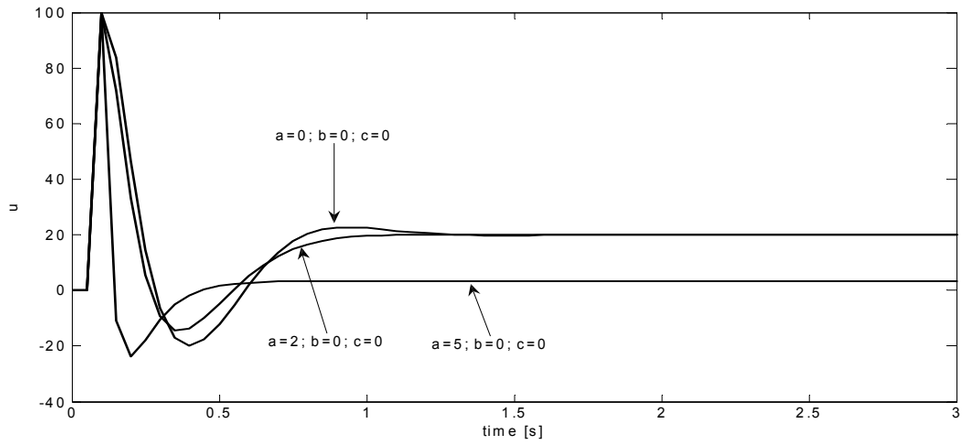

Fig. 7. Time responses of control value for various values of the parameter *a* in (8)

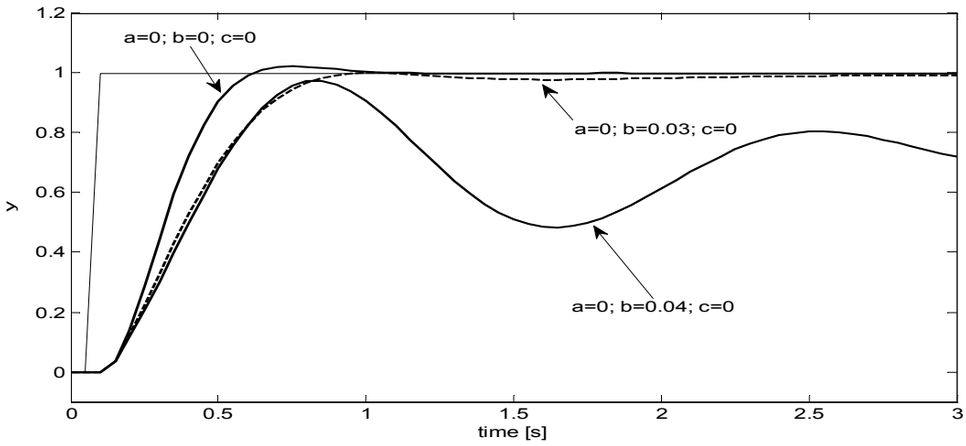

Fig. 8. Closed-loop time responses for various values of the parameter *b* in (8)

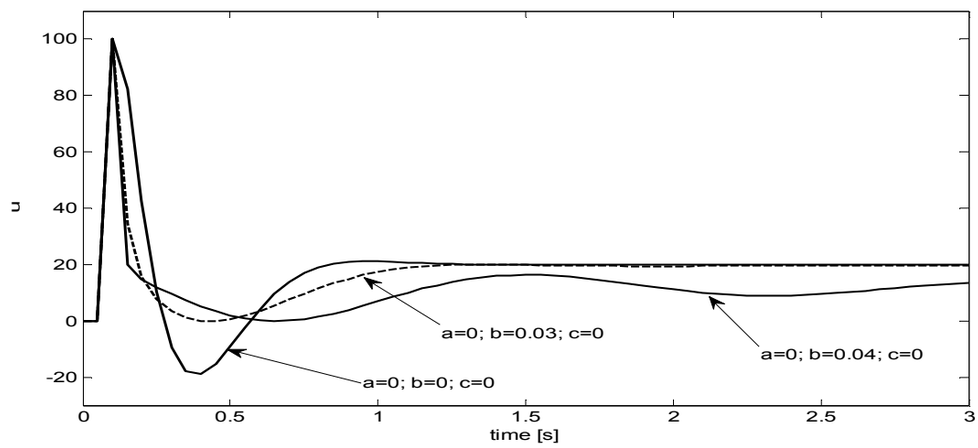

Fig. 9. Time responses of control value for various values of the parameter *b* in (8)

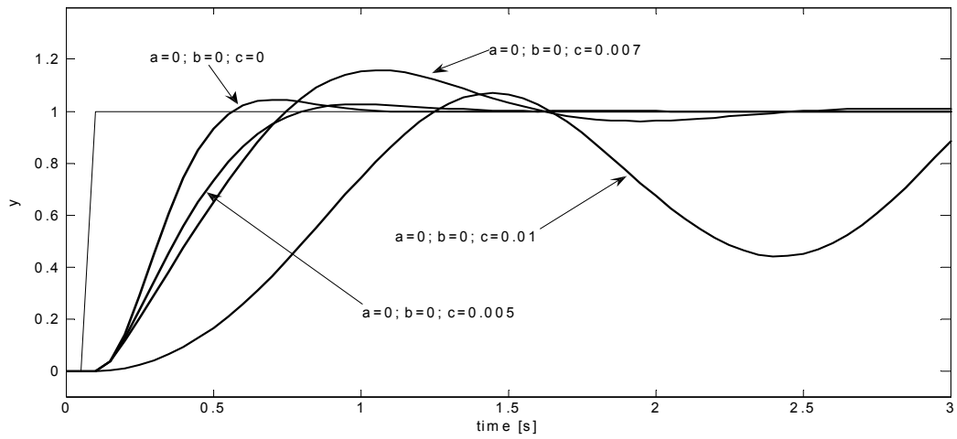

Fig. 10. Closed-loop time responses for various values of the parameter *c* in (8)

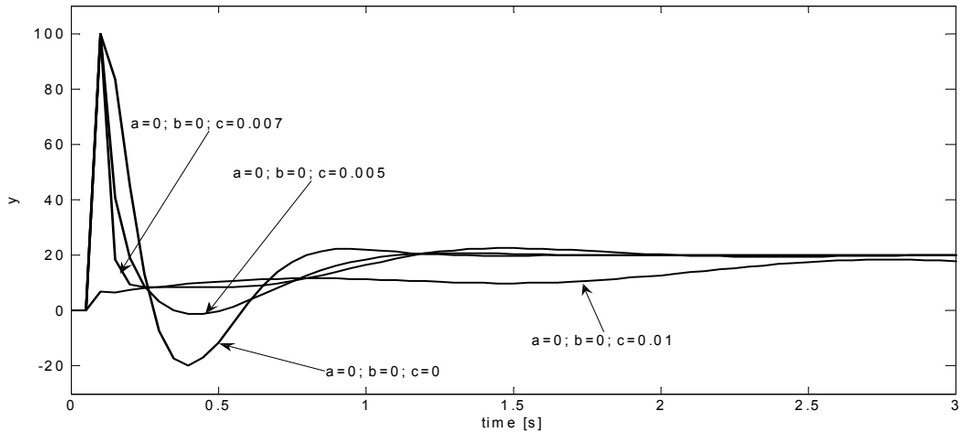

Fig. 11. Time responses of control value for various values of the parameter *c* in (8)

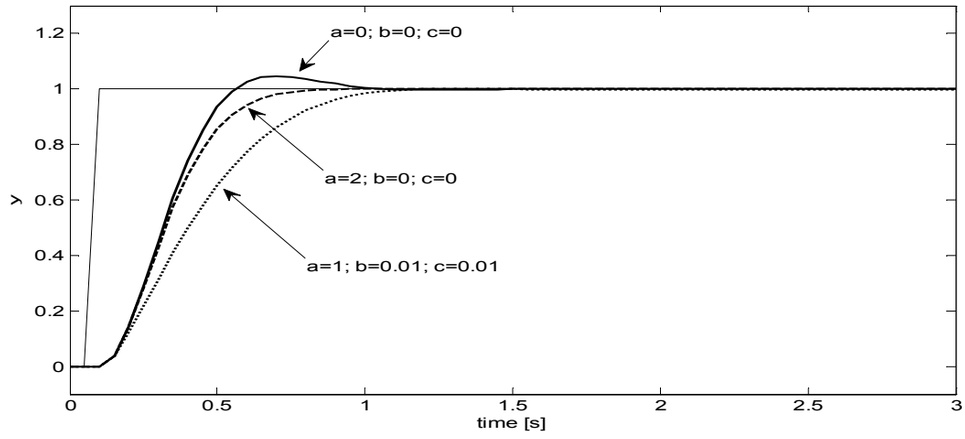

Fig. 12. Closed-loop time responses for various values of the parameters *a,b,c* in (8)

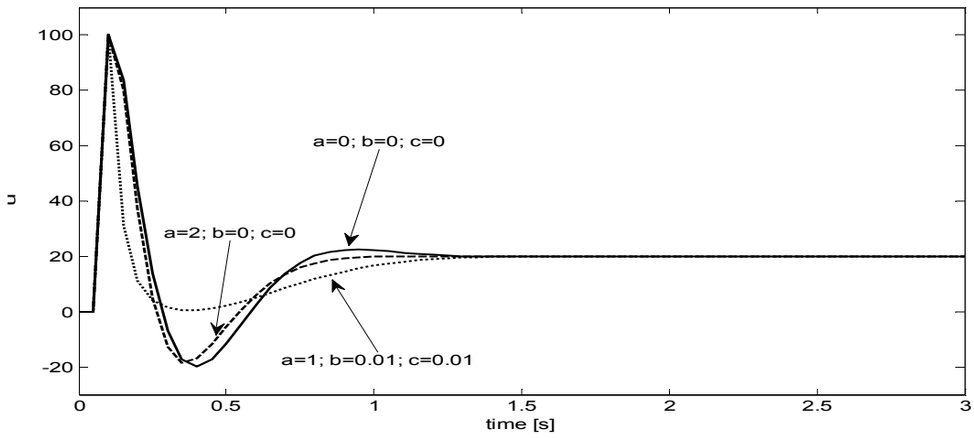

Fig. 13. Time responses of control value for various values of the parameter *a,b,c* in (8)

3. Robust controller design

3.1 Design in fixed defined working points

Consider $c = \{c_1, c_2, \dots, c_q\}$ to be the set of designed controller parameters and let the $s = \{s_1, s_2, \dots, s_r\}$ is the set of parameters of the controlled system. During the operation of the plant, the parameters s_i can move within some uncertainty space

$$S: \quad s_{i,\min} \le s_i \le s_{i,\max} \; ; \quad i = 1,2,...,r \tag{10}$$

where $s_{i,min}$ and $s_{i,max}$ are the minimum and maximum possible values of the *i*-th system parameter, respectively. Consider *W* different (physical) working points of the controlled process, defined by different vectors *s*, which are to be controlled by the robust controller. For that case consider the cost function in the additive form

$$J = \sum_{i=1}^{W} J_i \tag{11}$$

comprising performance evaluation (for instance (1)) in all *W* working points. It is also recommended to include the measured noise from the real system or other possible disturbances or expected situations in the simulation model. Note, that alternatively to the set of *W* defined physical working points we can use a set of 2^r system parameter vectors located in the vertices of a polytope representing bounds of the parameter space *S*.

3.2 Random generating of working points

Alternative to the previous method, for the working point selection the following method can be considered (Sekaj & Šrámek, 2005). In each generation of the GA, *n* random working points (for all chromosomes of the population the same ones) are generated i.e. *n* vectors (say *n=100*) of system parameters *s* become random values from the space *S*. For each individual of the population the fitness function is calculated using the performance index (11). In the next generation other *n* random parameter vectors are generated. In each generation the cost function evaluation is as follows:

1. random generation of n system parameter vectors s (working points),
2. closed-loop simulation and evaluation of the performance index for each individual and each working point,
3. evaluation of the cost function (11) for each individual.

3.3 Experimental comparison of robust design methods

To demonstrate the proposed controller design methods, consider the controlled system, which is described by the non-linear differential equation

$$y'' + a_2 y' + a_1 y + a_0 y^3 = b_0 u \tag{12}$$

The gain of this system is not constant and depends on the system output value y. Let the system parameters move within the uncertainty intervals

$$b_0 \in (1;5), a_0 \in (0.02;5), a_1 \in (0.02;0.2), a_2 \in (0.01;20) \tag{13}$$

Consider a PID controller described by the transfer function

$$G_{PID}(s)=P+I/s+Ds \tag{14}$$

with the control variable limited within the range $-10 \le u \le 10$. The parameters P, I and D have been designed using the following 4 methods:

Method 1: Robust approach according to section 3.2 with 100 randomly generated working points from the space S in each generation.

Method 2: Simple PID design using GA according to section 2, eq. (1) in the nominal working point (WP_3 in next method).

Method 3: Robust approach according to the IMC+PID method (Rivera et al., 1986) designed in 3 fixed working points:

WP_1: $a_0=0.02$; $a_1=0.02$; $a_2=0.01$; $b_0=1$ - lower limits of intervals (13)

WP_2: $a_0=5$; $a_1=0.2$; $a_2=20$; $b_0=5$ - upper limits of (13)

WP_3: $a_0=2.51$; $a_1=0.11$; $a_2=10.005$; $b_0=3$ - mean values of (13), nominal working point.

Method 4: Conventional "Optimal Module" PID design method (Oldenburg & Sartorius, 1956) using a linearised system in the nominal working point.

The design results are summarised in Table 1.

Method	P	I	D	SRM
1	100	12.62	4.41	0.2898
2	98.03	8.66	8.45	0.3521
3	89.10	15.41	2.92	0.3058
4	20.56	0.98	0.36	1.0124

Table 1. Results of the Robust PID design methods

The proposed *Statistical robustness measure* (SRM) quantifies statistical evaluation of a set of closed-loop simulation experiments (in our case 1200) with randomly generated working points from the uncertainty space S. The SRM can be expressed by a scalar value calculated as follows

$$SRM = \frac{1}{N}\sum_{i=1}^{N} J_i \tag{15}$$

where N is the number of closed-loop simulation experiments and J is a selected performance index. In our case (1) has been used.
All obtained results from Tab.1 have been tested on 25 randomly selected working points from the parameter space S (all 25 are identical for all 4 design methods). Time-responses for all design methods are depicted in Fig. 14-Fig. 17.

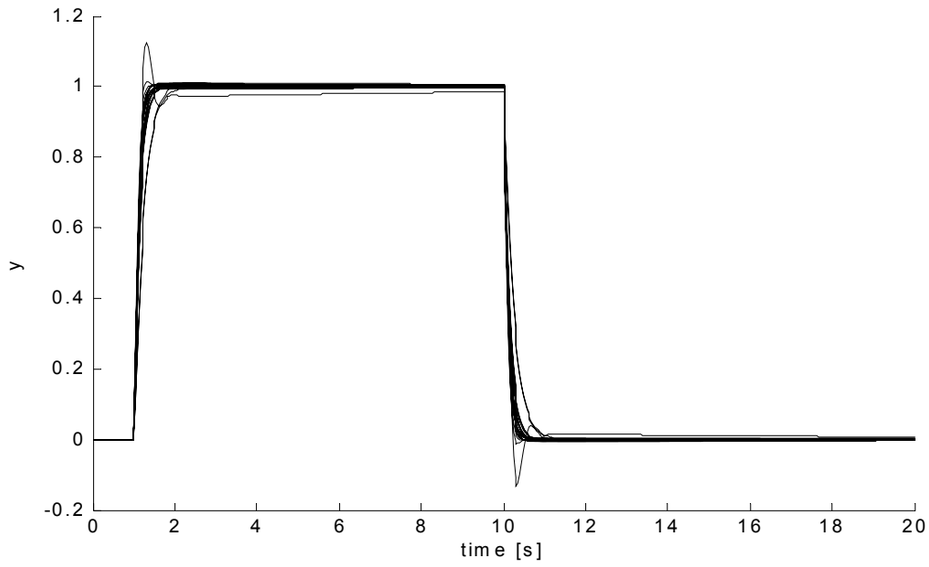

Fig. 14. Closed-loop time responses of 25 test systems for Method 1

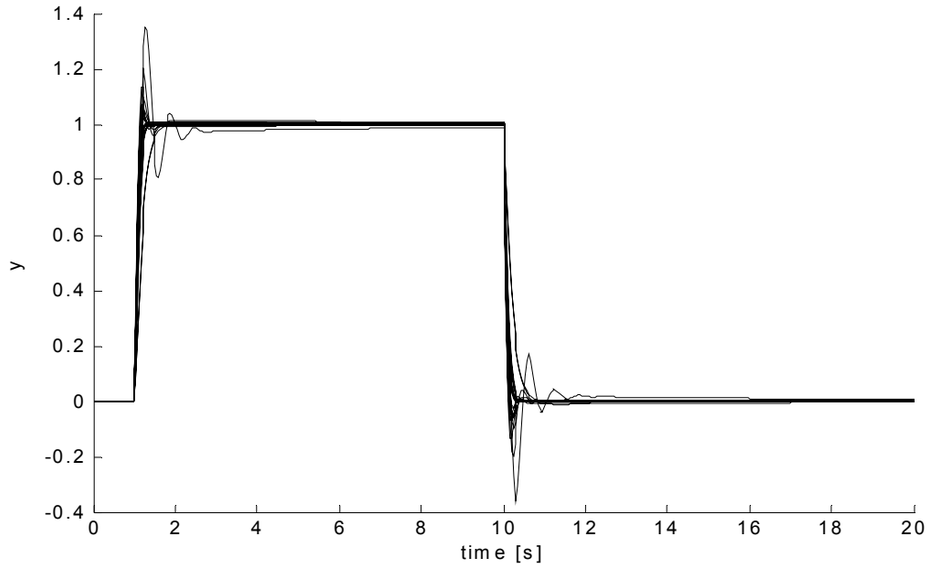

Fig. 15. Closed-loop time responses of 25 test systems for Method 2

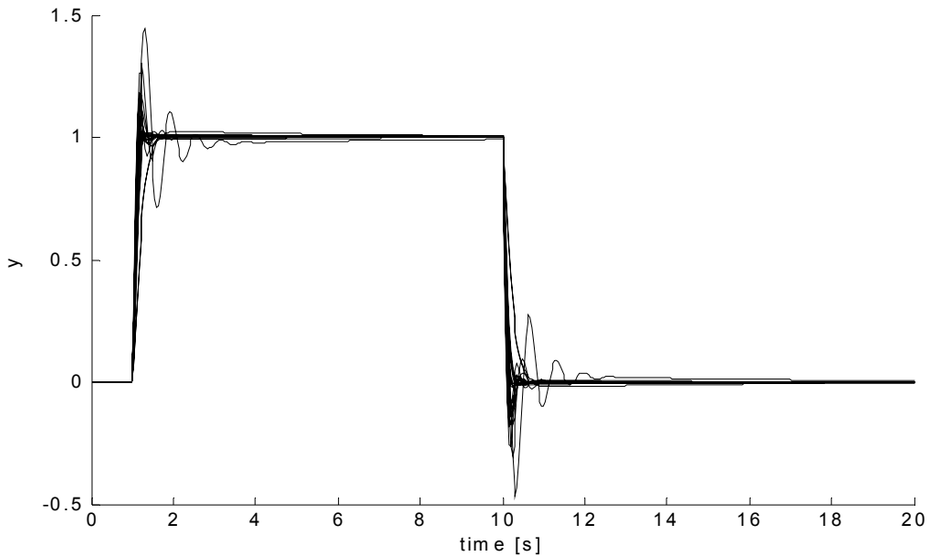

Fig. 16. Closed-loop time responses of 25 test systems for Method 3

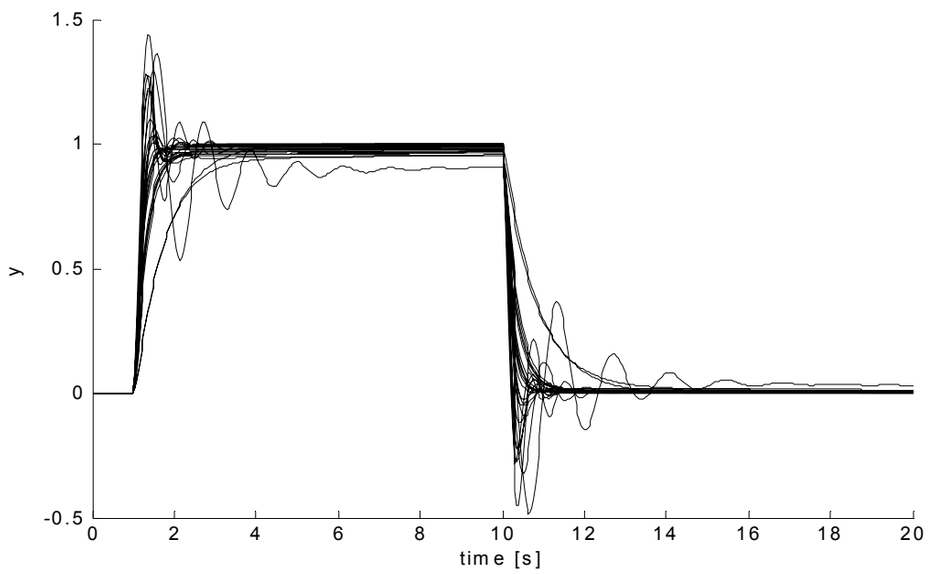

Fig. 17. Closed-loop time responses of 25 test systems for Method 4

4. Multi-objective controller design

4.1 Design principle

In solving many practical design problems not just a single optimisation objective is considered. Moreover, the particular objectives are in contradiction (e.g. performance vs. energy consumption, etc.). A common way of solving multi-objective tasks is using a single cost function consisting of multiple parts

$$J = w_1 f_1 + w_2 f_2 + \ldots + w_n f_n \tag{16}$$

where each part f_i, $i = 1, ..., n$ represents an objective with some weight w_i. The main disadvantage of this method is the high sensitivity of the solution to the weight coefficients. This can lead to solutions, which do not correspond to our primary requirements.
Another way for solving multi-objective problems is the use of the Dominance principle, which is the search for the Pareto-optimal set of solutions. This is an effective way to overcome the above mentioned problem. Several authors have used this approach in solving various multi-objective problems e.g. (Corne et al., 2000; Fleming & Purshouse; Zitzler & Thiele, 1998). Consider minimisation problem. According to the Dominance principle the individual x dominates the individual y (or the individual y is dominated by the individual x) if

$$\begin{aligned} &\forall i = 1,2,...,n\ ;\ f_i(x) \le f_i(y) \text{ and} \\ &\exists j; j \in \{1,2,...,n\}\ ;\ f_j(x) < f_j(y) \end{aligned} \tag{17}$$

where n is the number of objectives and f_i, $i=1,2,...,n$ is the cost function corresponding to the i-th objective. In case of maximisation tasks the formulation is analogical. A set of individuals, which are non-dominated by another individual are members of the Pareto-optimal set of solutions.
Now, the objective is not to find a single solution, but as much as possible non-dominated solutions, for which it is not possible to decide, which one is better. Each user will select the individual, which is the best with respect to his requirements. Based on the above approach the search algorithm is as follows:

1. Generating of the initial population.
2. Calculation of all objective functions for each individual of the population.
3. Domination calculation: each individual of the population will obtain such a number of "penalty points", which corresponds to the number of individuals by which it is dominated. The number of penalty points represents the final minimised cost function.
4. Individuals with zero penalty points are stored in the current group of non-dominated individuals.
5. Calculation of the new population (selection, crossover, mutation).
6. Adding the current non-dominated group to the new population.
7. Testing of terminating conditions, jump to Step 2 or end.

In case of controller design applications various objectives can be considered: integral performance indices (as in section 2), settling time, maximum overshoot, oscillation damping, various stability measures, gain/phase margin, energy consumption, other economic aspects, minimisation of negative environmental impacts of the controlled process operation, etc.

4.2 Case study

Consider the design of a DC motor speed controller. The objectives are short settling time and, on the other hand, low control energy consumption. To fulfil the first objective let us minimise the simple integral criterion (1).
The second objective can be represented by minimisation of the integral of the input motor voltage square, which is proportional to the input energy consumption

$$J_2 = \int_0^T u^2(t)dt \tag{18}$$

The results obtained using the algorithm described in the section 4 are depicted in Fig.18 (J_2 versus J_1). Individuals, which have occurred during the GA run are marked by "x". The non-dominated solutions from the last generation are in the left bottom part of the area marked with "o". Detail of the non-dominated set is also depicted in Fig.19. The individual marked "*Min(J1)*" represents the solution with the best performance with respect to the objective J_1 and the individual marked "*Min(J2)*" represents the best solution with respect to the objective J_2. The individual marked "*Compromise*" is a selected trade off between the both previous extremes. Step responses for all three mentioned solutions are depicted in Fig.20 and Fig.21.

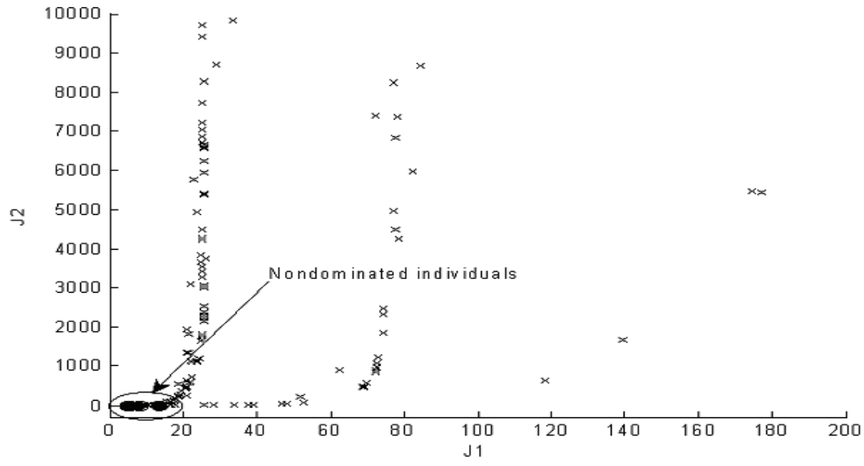

Fig. 18. Individuals occurred during the multi-objective evolution

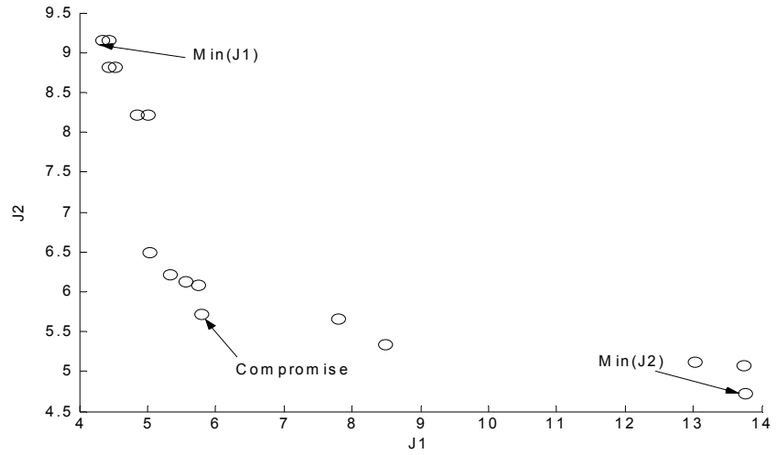

Fig. 19. Set of non-dominated individuals (Pareto-optimal set) - detail of Fig.18

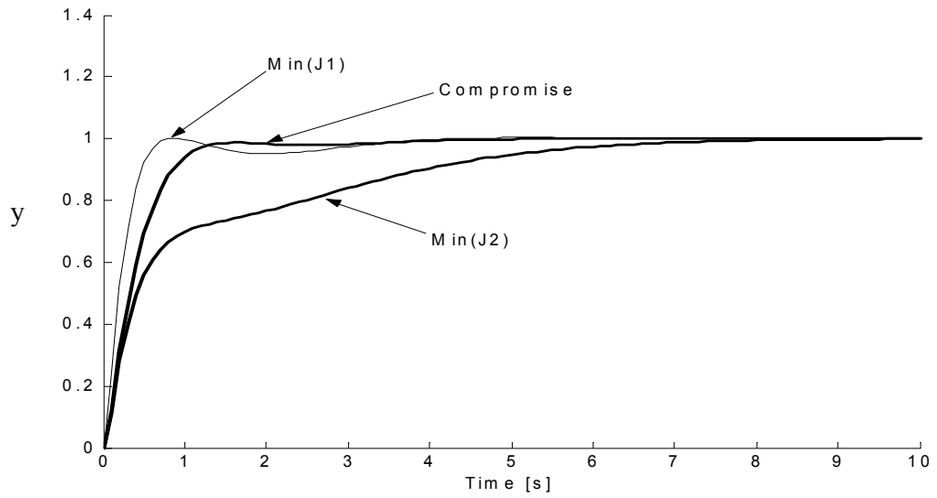

Fig. 20. Time-responses of three selected individuals

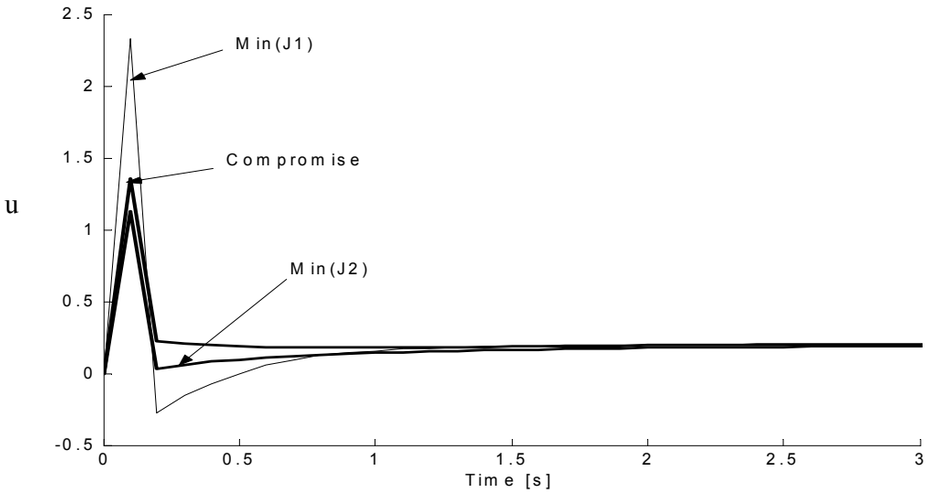

Fig. 21. Control variable time-responses of the three selected individuals

5. Controller internal structure design

In the previous parts the design goal was the optimisation of defined controller parameters. In this section also the internal structure of the controller is designed/optimised. Because the structure of the controller is unknown, also the number of its parameters is not defined. From that reason, the length of the chromosome is changing. For such a task the Genetic Programming has been used (Koza, 1992; Banzhaf et al., 1999). Next two various approaches are proposed (Sekaj et al., 2005; Sekaj & Perkacz, 2007) . In the first approach a discrete-time recurrent control algorithm is considered. In the second an interconnected network of elementary dynamic and static building blocks are used for a controller design.

5.1 Discrete-time controller algorithm design

In this proposed approach a discrete-time recurrent control algorithm has been designed as function of defined time-delayed input variables (Sekaj et al., 2005). The following variables have been considered: *e(k), e(k-1), ..., e(k-m), y(k), y(k-1), ..., y(k-n), u(k-1), u(k-2), ..., u(k-p), r(k),* where *e* is the control error, *y* is the controlled output, *u* is control value and *k* is the control step. The aim is to find the optimal form of the controller function

$$F(\theta)=? \qquad \theta=\{e(k),..., e(k-m), y(k),..., y(k-n), u(k-1),..., u(k-p),r(k),c\} \tag{19}$$

such that the cost function (2) is minimised, where *c* are real constants. For the representation of the function *F* the in GP commonly used tree representation has been used. The functional nodes contain operations +, -, * and the terminating nodes include operation arguments from the vector θ. An example of such a tree is in Fig. 22.

Following genetic operations have been applied to modify the tree-represented individuals. The first one is the crossover, which is the exchange of randomly selected sub-trees in two parent trees (see Fig. 23a). From the mutation operators (unary operations) the following have been used: replacement of a randomly selected sub-tree by another randomly generated sub-tree, removal of a randomly selected sub-tree or addition of a randomly generated sub-tree (see Fig. 23b). The last mutation type is the mutation of a terminating node, which is a random change of a constant (or a variable) to another value or another variable.

Let us demonstrate the design approach on a controller design for a simple oscillatory linear system with the transfer function

$$G(s)=\frac{1}{s^2+0.001s+1} \tag{20}$$

The cost function in form (2) has been used. After 3000 generations the following recurrent control algorithm has been obtained

$$\begin{aligned} u(k) = 6.74([y(k) - y(k\text{-}2) - e(k)][e(k\text{-}2) - 10.88] [23.25 + y(k\text{-}1) + y(k\text{-}2)]) - -87.82e(k\text{-}2) \\ [9.33 + y(k\text{-}2)] + y(k) + e(k) - e(k\text{-}1) \end{aligned} \tag{21}$$

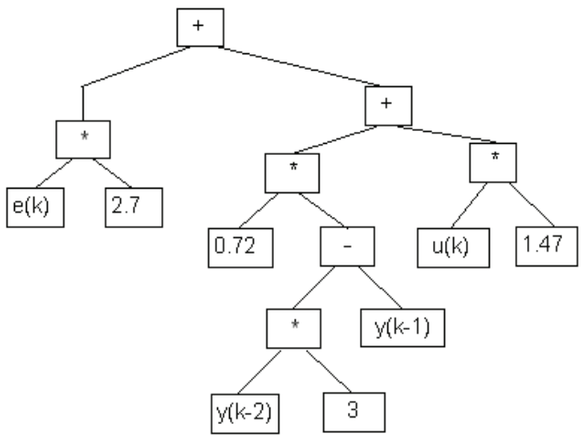

Fig. 22. Example of a controller function tree representation

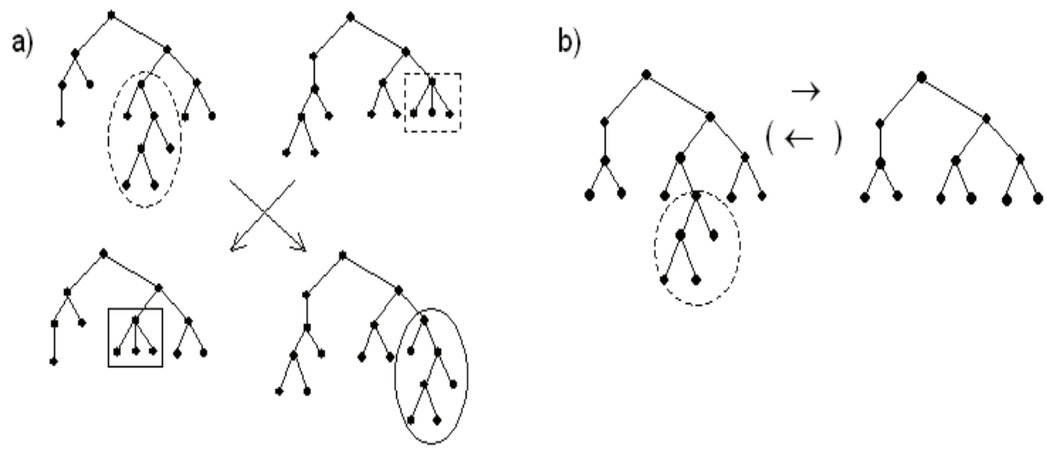

Fig. 23. a) crossover of two trees, b) mutation by removing (adding) of a sub-tree

In Fig. 24 the closed-loop response under the obtained solution after a reference signal step and an external disturbance is compared with a time response under a PID designed using GA with the same cost function.

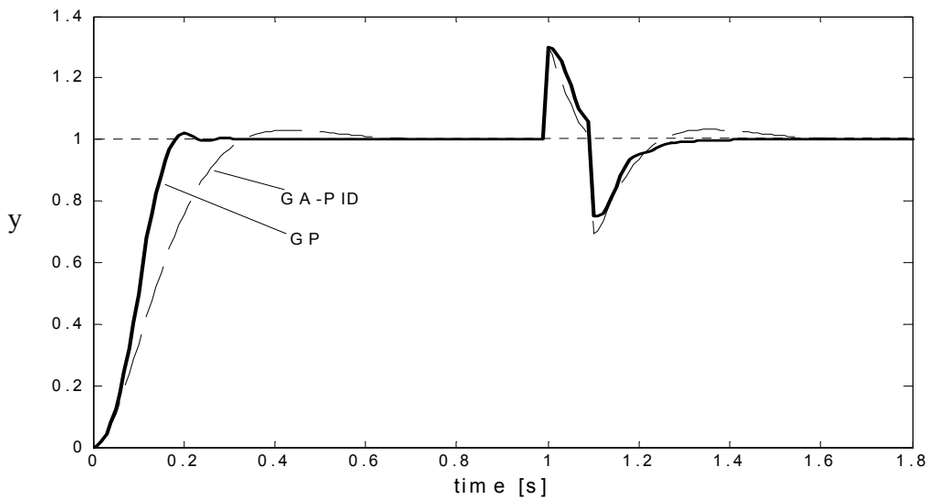

Fig. 24. Time responses with GP and GA designed controllers

5.2 Interconnected controller network

This controller representation is based on an interconnected network, which consists of the following elementary continuous-time dynamic/static function blocks: integrator, derivative unit, amplifier (multiplication by a constant) and a summation/multiplication unit (see Fig. 25) where *A, B* and *D* are real constants (Sekaj & Perkacz, 2007). The objective is to find the optimal control network consisting of such elementary function blocks and their interconnections, which minimises the cost function (1) or (2).

To represent each potential solution the following table or matrix format of a chromosome has been used (see example in Table 2). Each column represents one of the above-mentioned blocks. The second row includes the type of each block (Mul-multiplication, Sum-

summation, Der-derivative, Int-integral). The third row contains the number of the next block connected to its output. The number *0* indicates the controller output. The fourth row indicates the number of the previous block or signal connected to the block input. Negative numbers indicate the controller input signals, positive numbers are function blocks. It is possible to reduce the coding algorithm in such a way, that for each block it is sufficient to use only one input and one output. The last row contains a multiplicative constant of each block. Summation blocks need not be present in the table. For demonstration of the coding mechanism, the genotype in Table 2 corresponds to the controller in Fig. 26. In our case *-1* represents the control error signal *e*.

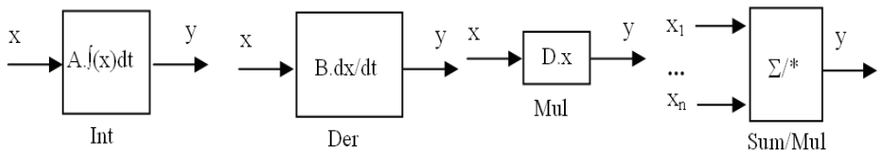

Fig. 25. Library of the used elementary function blocks

block	1	2	3	4	5
attribute	Mul	Mul	Der	Mul	Mul
	0	0	2	0	0
	-1	3	-1	-1	4
	36.92	3.57	31.96	36.08	21.54

Table 2. Chromosome representation

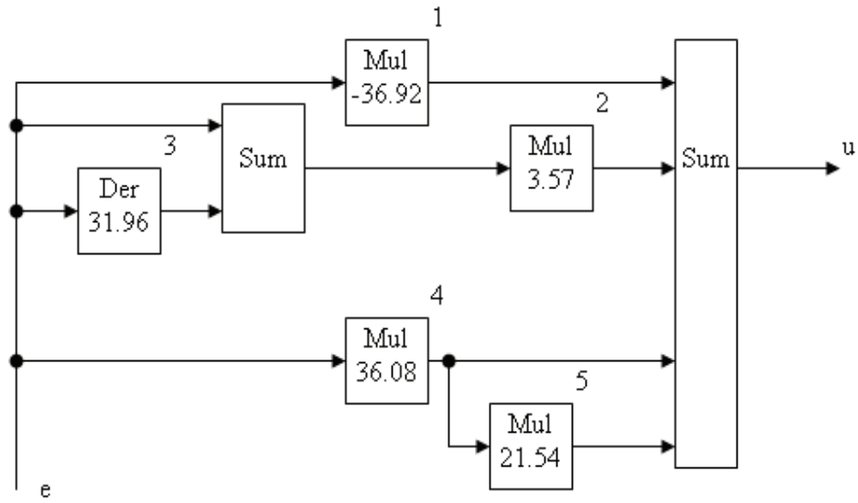

Fig. 26. Controller network example

Specialised crossover and mutation operators have been programmed to modify chromosomes. In case of crossover, two selected parent chromosomes are divided, both in random positions between two columns of the table, and then their appropriate parts are exchanged. For mutation the following operations have been used: the first is the random change of the block type, i.e. a block (for example integrator) is modified to another block type (e.g. a derivative unit). A next mutation type is the random deletion or addition of a block in the scheme. The last used mutation type is the random change of a constant, which

is a part of each block type. After each modification some connections may be missing or excessive. Therefore, the unnecessary ones are removed or the new ones are randomly added and connected to randomly selected points.
The design approach has been verified on the controller design for the linear dynamic system

$$G(s) = \frac{s^2 + 3s + 1}{4s^5 + 8s^4 + 10s^3 + 6s^2 + 3s + 1} \tag{22}$$

The objective was to minimise the performance index (2). The designed controller structure which was obtained after 3000 generations with the population size 80 individuals, mutation rate 0.3 and the crossover rate 0.3 is in Fig. 27. The closed-loop response of the obtained controller (GP) to the reference signal is in Fig. 28 compared with PID controller, which was optimised using GA (GA-PID, according section 2).

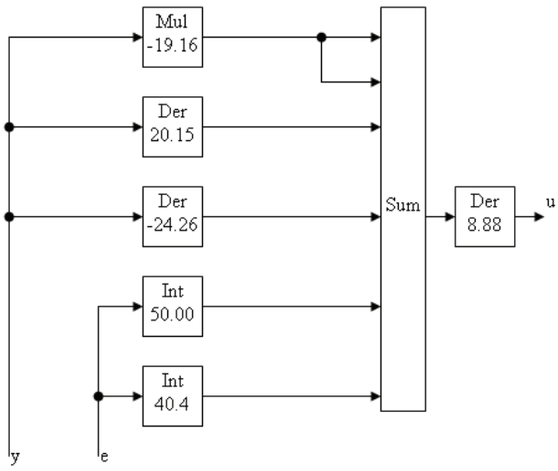

Fig. 27. Controller structure

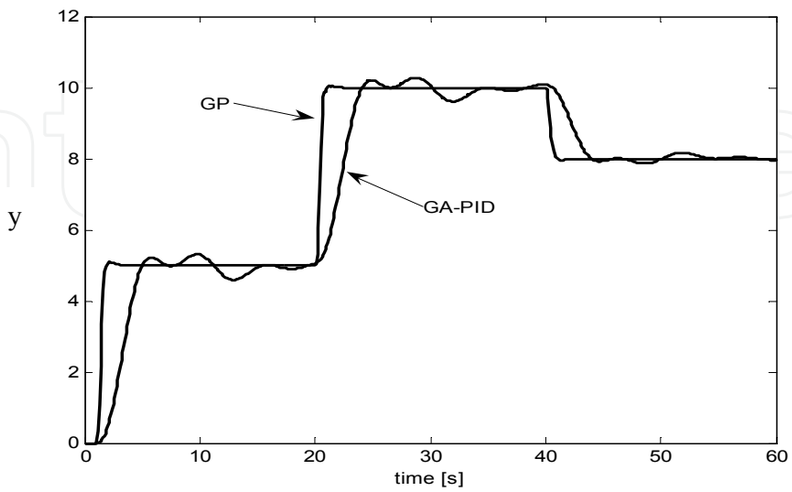

Fig. 28. Time response of the designed controller for reference signal steps from the value 0 to 5, 10 and 8

6. Conclusion

In this chapter evolutionary based design/optimisation approaches has been proposed for controller design of continuous-time process control. Parameters of controllers with fixed defined internal structure are designed as well as controllers with a-priori unknown internal structure and its parameters. The presented approaches minimise a cost function, which comprises closed-loop system simulation and performance index evaluation. In this way the controller design is transformed into a search problem in the *n*-dimensional parameter space.

The design/optimisation can be carried out for complex systems and control structures of various types. The main and practically the only limitation of the approach is the time-consuming computation (compared with conventional approaches) due to thousands up to ten thousands closed-loop simulations needed by each design procedure. From the point of view the user, on the other hand, the design method is simple to use. It transfers the design effort from the experienced human designer to the computer.

The design approach is simple to extend to robust controller design, for which two different methods have been proposed. In addition statistical robustness measure has been introduced, which can be considered as an objective tool for robust controller performance comparison. Next, the design idea has been extended also to a multi-objective design task, where the objective is the search for the Pareto-optimal set of solutions. From these solutions the designer can choose the representative, which is the most appropriate in the particular case.

Finally the design goal was extended from a fix defined controller structure with unknown controller parameters to the search/optimisation of the unknown internal structure of the controller. From that reason, the Genetic Programming has been used.

The proposed evolutionary-based methods can be used for design of various types of controllers for various system types (linear, non-linear, stable, unstable, SISO and MIMO, fuzzy, neural, etc.). The only condition of this approach is that there exists a simulation model of the designed closed-loop.

In the future, this design approach will be extended for solving very complex design tasks in the process control area like complex MIMO control systems for non-linear continuous-time systems and for robotic applications using parallel evolutionary algorithms.

7. References

Banzhaf, W.; Nordin, P.; Keller, R. E.; Francone, F. D. (1999). *Genetic Programming: An introduction,* Morgan Kaufmann, San Francisco

Corne, D.W.; Knowles, J.D.; Oates, M.J. (2000). Pareto Enveloped-based Selection Algorithm, In: *Proceedings of the Conference Parallel Problem Solving from Nature VI,* Springer, pp. 839-848

Dorf, R. C. (1990). *Modern Control Systems,* Addison-Wesley publishing Company, 5th edition

Eiben, A. E.; Smith, J. E. (2003). *Introduction to Evolutionary Computing*, Springer

Fleming, P. J.; Purshouse, R. C. Evolutionary Algorithms in Control System Engineering: A Survey, In: *Control Engineering Practice,* 10(11), 1223

Fonseca, C.M.; Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimisation: Formulation, Discussion and Generalisation, In: *Proceedings of the Conference on Genetic Algorithms,* San Mateo, CA, Morgan Kaufmann

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addisson-Wesley

Grosman, B.; Lewin, D.R. (2005). Automatic Generation of Lyapunow Functions Using Genetic Programming, In: *16th World Congress of IFAC*, Prague, July 3-8

Herrero, J.M.; Blasco, X.; Martínez, M.; Salcedo, J.V. (2002). Optimal PID Tuning with Genetic Algorithms for Non Linear Process Models, In: *Proceedings on the 15th World Congress of IFAC,* Barcelona, July 21-26

Kawabe, T.; Tagami, T.; Katayama, T. (1996). A Genetic Algorithm based Minimax Optimal Design of Robust I-PD Controller, In: *UKACC Int. Conference on Control '96,* Conf. Publication No. 427, IEE, pp.436-441

Khatib, W.; Silva, V.; Chipperfield, A.; Fleming, P. (1999). Multidisciplinary Optimisation With Evolutionary Computing for Control Design, In: *Proceedings on the 14th World Congress of IFAC,* Beijing, July 5-9

Koza, J.R. (1992). *Genetic Programming*, Cambridge, MA, MIT Press

Koza, J.R.; Yu, J.; Keane, M.A.; Mydlowec, W. (2000). Evolution of a controller with a free variable using genetic programming. In: *Proceedings on the European Conference EuroGP 2000*, Edinburgh, Scotland, UK, Lecture Notes in Computer Science, Volume 1802. Berlin, Germany: Springer-Verlag, pp. 91–105, ISBN 3-540-67339-3, April 2000

Krohling, R.A.; Rey, J.P. (2001). Design of Optimal Disturbance Rejection PID Controllers Using Genetic Algorithms, In: *IEEE Trans. On Evolutionary Computation,* Vol.5, No.1

Kuo, B.C. (1991). *Automatic Control Systems,* Prentice-Hall International Editions

Lewin, D.R. (2005). Evolutionary Algorithms in Control System Engineering, In: *Proceedings on the 16th World Congress of IFAC*, July 3-8, Prague

Man, K.F.; Tang, K.S.; Kwong , S. (2001). *Genetic Algorithms, Concepts and Deign,* Springer

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolutionary Programs,* Springer

Mitsukura, Y.; Yamamoto, T.; Kaneda, M.; Fujii, K. (1999). Evolutionary Computation in Designing a PID Control System, In: *Proceedings on the 14th IFAC World Congress,* Beijing, 5-9 july, P.R.China, pp. 497-502

Molina-Cristóbal, A.; Griffin, I.A.; Fleming, P.J.; Owens, D.H. (2005). Multiobjective Controller Design: Optimising Controller Structure with GA, In: *Proceedings on the 16th World Congress of IFAC*, July 3-8, Prague

Oldenburg, L.C.; Sartorius, H. (1956). A uniform approach to the optimum adjustments of control loops, *Frequency response*, The MacMillan Co., N.Y.

Rivera, D.E.; Morari, M.; Skogestad, S. (1986). Internal model control for PID controller design, In: *Ind. Eng. Chem. Res. 25* (1), pp. 252–265

Sekaj, I. (1999). Genetic Algorithm-based Control System Design and System Identification, In: *Proceedings on the Int. Conference Mendel'99*, June 9-12, Brno, Czech Republic, pp.139-144

Sekaj, I.; Foltin, M.; Gonos, M. (2002). Genetic Algorithm Based Adaptive Control of an Electromechanical MIMO System, In: *Proceedings of the GECCO Conference*, July 9-13, New York, pp. 696

Sekaj, I.; Veselý, V. (2005). Robust output feedback controller design: genetic algorithm approach, In: *IMA Journal of Mathematical Control and Information,* 22, pp. 257-263

Sekaj, I. (2003). Genetic Algorithm Based Controller Design, In: *2nd IFAC conference Control System Design'03*, Bratislava, Slovak Republic, September 7-10

Sekaj, I.; Perkácz, J.; Nídel, M. (2005). Controller design based on genetic programming, In: *Proceedings of the Int. conference Mendel'05*, Brno, Czech Republic, June 15-17

Sekaj, I.; Šrámek, M. (2005). Robust Controller Design Based on Genetic Algorithms and System Simulation, In: *Proceedings on the conference CDC-ECC'05*, Seville, Spain, December 12-15

Sekaj, I.; Perkácz, J. (2007). Genetic Programming Based Controller Design, In: *Proceedings of the IEEE Congress on Evolutionary Computation'07*, Singapore, September 25-28Sweriduk, G.D.; Menon, P.K.; Steinberg, M.L. (1999). Design of a pilot-activated recovery system using genetic search methods, In: *Optimal Synthesis*

Yang, Z.Y.; Chan, C.W.; Xue, M.S.; Luo, G.J. (2005). On-line Temperature Control of an Oven Based on Genetic Algorithms, In: *Proceedings on the 16th World Congress of IFAC*, Prague, July 3-8

Zitzler, E.; Thiele, L. (1998). Strength Pareto Evolutionary Algorithm, In: *Technical Report 43, Computer Engineering and Communication Networks Lab (TIK),* Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998

Evolutionary Computation

Edited by Wellington Pinheiro dos Santos

ISBN 978-953-307-008-7
Hard cover, 572 pages
Publisher InTech
Published online 01, October, 2009
Published in print edition October, 2009

This book presents several recent advances on Evolutionary Computation, specially evolution-based optimization methods and hybrid algorithms for several applications, from optimization and learning to pattern recognition and bioinformatics. This book also presents new algorithms based on several analogies and metafores, where one of them is based on philosophy, specifically on the philosophy of praxis and dialectics. In this book it is also presented interesting applications on bioinformatics, specially the use of particle swarms to discover gene expression patterns in DNA microarrays. Therefore, this book features representative work on the field of evolutionary computation and applied sciences. The intended audience is graduate, undergraduate, researchers, and anyone who wishes to become familiar with the latest research work on this field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ivan Sekaj (2009). Evolutionary Based Controller Design, Evolutionary Computation, Wellington Pinheiro dos Santos (Ed.), ISBN: 978-953-307-008-7, InTech, Available from: http://www.intechopen.com/books/evolutionary-computation/evolutionary-based-controller-design

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen